0.1 Algorithmic

- Handout by Cody Johnson

Stuck? Try These: When to use algorithms

Rules of thumb to applying algorithms:

1. A procedure or operation is given. An algorithm would be a helpful organization of a set of allowed steps
2. You seek a construction. An algorithm is a powerful method to construct a configuration satisfying some given conditions
3. A complex object or configuration is given that you want to reduce.
4. You are looking to represent all positive integers n in some given form. Often you are able to represent the items greedily.
5. You are packing anything. Often you are able to pack the items greedily.
6. You are looking for a winning strategy. A winning strategy is, by definition, an algorithm to ensure a player of winning a game.

0.1.1 Data Structures

Data Structure (Binary Heap)— Segment tree where, node `n` has children `2n+1, 2n+2`.

1 `timus 1862`, Sum of operations

Data Structure (Disjoint Set)— This data structure keeps track of connectivity by assigning a representative to a connected subset. And for any two nodes, `u, v` they are connected iff they have the same representative.

0.1.2 Minimal Spanning Tree

Definition (Minimum Spanning Tree)— A minimum spanning tree or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted (un)directed graph that connects all the vertices's together, without any cycles and with the minimum possible total edge weight.

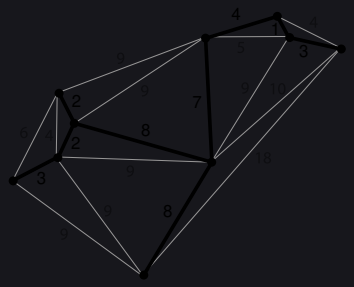

Figure 0.1: Minimal Spanning Tree

Lemma 0.1.1 (MST Cut) — For any cut of the graph, the lightest edge in that cut-set is in evey MST of the graph.

Proof. Assume that there is an MST T that does not contain e. Adding e to T will produce a cycle, that crosses the cut once at e and crosses back at another edge e'. Deleting e' we get a spanning tree $T/e' \cup e$ of strictly smaller weight than T. This contradicts the assumption that T was a MST.

Lemma 0.1.2 (MST Cycle) — For any cycle C in the graph, the heaviest edge in the cycle cannot be in any MST of the graph.

Proof. Assume the contrary, i.e. that e belongs to an MST T_1. Then deleting e will break T_1 into two subtrees with the two ends of e in different subtrees. The remainder of C reconnects the subtrees, hence there is an edge f of C with ends in different subtrees, i.e., it reconnects the subtrees into a tree T_2 with weight less than that of T_1, because the weight of f is less than the weight of e.

Algorithm (Kruskal's Algorithm)— Kruskal's algorithm is a 'minimum-spanning-tree algorithm' which finds an edge of the least possible weight that connects any two trees in the forest. It is a greedy algorithm in graph theory as it finds a minimum spanning tree for a connected weighted graph adding increasing cost arcs at each step.

Solution. To optimize this algorithm, Disjoint Set DS is used.

Algorithm (Prim's Algorithm)— Greedily build the tree by adding edges one by one. At one step we add the minimal cost edge that connects the tree to the vertices's that are not in the tree.

0.1.3 Shortest Path Problem

Definition (Shortest Path Problem)— Finding the shortest path between two nodes in a weighted or unweighted graph.

Algorithm (Breadth-First Search)— This algo runs from a node and "levelizes" the other nodes.

Algorithm (Dijkstra's Algorithm)— It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbor, and updates the neighbor's distance if smaller.

0.1.4 Other CP Tricks

> **Theorem 0.1.3** (Swap Sort) — In any swap sorting algorithm, the number of swaps needed has the same parity.

Algorithm (Convex Hull Trick)— Given a lot of lines on the plane, and a lot of queries each asking for the smallest value for y among the lines for a given x, the optimal strategy is to sort the lines according to their slopes, and adding them to a stack, checking if they are relevant to the 'minimal' convex hull of those lines.

Notice that the line y=4 will never be the lowest one, regardless of the x-value. Of the remaining three lines, each one is the minimum in a single contiguous interval (possibly having plus or minus infinity as one bound), which are colored green in the figure.

Thus, if we remove "irrelevant" lines such as $y = 4$ in this example and sort the remaining lines by slope, we obtain a collection of N intervals (where N is the number of lines remaining), in each of which one of the lines is the minimal one. If we can determine the endpoints of these intervals, it becomes a simple matter to use binary search to answer each query.

Figure 0.2: Convex Hull Trick

0.1.5 Parity of Permutation

- Sign of a permutation - Keith Conrad

Definition (Permutation as Composition of Transpositions)— Any cycle in S_n is a product of transpositions: the identity (1) is $(12)(12)$, and a k -cycle with $k \geq 2$ can be written as

$$(i_1 i_2 \cdots i_k) = (i_1 i_k)(i_1 i_{k-1}) \cdots (i_1 i_3)(i_1 i_2)$$

Or

$$(i_1 i_2 \cdots i_k) = (i_{k-1} i_k) \cdots (i_2 i_k)(i_1 i_k)$$

For example, a 3 -cycle (abc) can be written as

$$(abc) = (bc)(ac) = (ac)(ab)$$

since any permutation in S_n is a product of cycles and any cycle is a product of transpositions, any permutation in S_n is a product of transpositions.

Theorem 0.1.4 (Parity of Transpostion Composition) — If the permutation σ is written as composition of two sets of tanspositions:

$$\sigma = \tau_1 \tau_2 \dots \tau_r = \tau'_1 \tau'_2 \dots \tau'_{r'}$$

Then $r \equiv r' \pmod 2$

Proof. We can write the identity permutation as the compostion of those transpositions:

$$(1) = \tau_1 \tau_2 \dots \tau_r \tau'_{r'} \dots \tau'_2 \tau'_1$$

We are done if we can prove that (1) can never be composed of an odd number of transpositions. Write (1) as:

$$(1) = (a_1, b_1)(a_1, b_2) \dots (a_k, b_k)$$

We prove by induction that k is even. Firstly notice that there must be another $i \neq 1$ such that $a_i = a_1$. Since

$$(ab)(cd) = (cd)(ab)$$

We can safely assume that $a_2 = a_1$. Now if $b_1 = b_2$, we can just remove $(a_1, b_1)(a_2, b_2)$ and by induction hold our claim. So suppose $b_1 \neq b_2$. Since

$$(ab)(ac) = (bc)(ab)$$

We can replace $(a_1, b_1)(a_2, b_2)$ by $(b_1, b_2)(a_1, b_1)$. So,

$$(1) = (b_1, b_2)(a_1, b_1) \dots (a_k, b_k)$$

But then there must be another $i \neq 1, 2$ such that $a_i = a_1$. And since this can't continue infinitely, we will eventually need to reduce k by 2. Which completes the induction.

Definition (Inversions of a Permutation)— Inversions of a permutation σ are pairs (x,y) such that

$$x < y \text{ and } \sigma(x) > \sigma(y)$$

Definition (Parity of a Permutation)— Let m be the number of inversions of a permutation σ, and let $\sigma = \tau_1\tau_2\dots\tau_r$ be a transposition composition of σ. Then the pairity of the permutation σ is defined by:

$$\operatorname{sgn}(\sigma) = (-1)^m = (-1)^r$$

Where the permutation is called *odd* if $\operatorname{sgn}(\sigma) = -1$ and *even* otherwise.

Parity of permutaions follow properties of integers under addition: even+even is even, odd+odd is even, even+odd is odd. Also the inverse of the polynomial has the same parity.

The parity can also be defined with polynomials. Let

$$P(x_1,x_2,\dots,x_n) = \prod_{i<j}(x_i - x_j)$$

Now for a given permutation σ of the numbers $\{1,\dots,n\}$, we can also define

$$\operatorname{sgn}(\sigma) = \frac{P\left(x_{\sigma(1)},\dots,x_{\sigma(n)}\right)}{P(x_1,\dots,x_n)}$$

Theorem 0.1.5 (Definitions of parity are equivalent) — The parity of the number of transpositions in the decomposition of σ is the same as the parity of the number of inversions.

Proof. Suppose that σ has r inverses and its decomposition into transpositions has m elements, that is

$$\sigma = \tau_1\tau_2\dots\tau_m$$

If we let $\sigma = a_1a_2\dots a_n$, and $a_k = 1$, then if we exchange a_{k-1}, a_k, that increases m by 1, and reduces r by 1. We can do this till σ becomes the identity, and the parity of $m-r$ would not change. But in that case, as we have proved in Theorem 0.1.4, $m-r$ is even.

Corollary 0.1.6 — The pairity of σ is also the pairity of the number of *even cycles* of σ.

Definition (Permutations as Matrices)— We can translate permutations using matrices. For $\sigma \in S_n$, let $T_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ by permuting the columns of I_n by σ. For example, (123) translates to

$$T_{(123)} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Then we have,

$$\det(T_\sigma) = \operatorname{sgn}(\sigma)$$

The mapping $\sigma \to T_\sigma$ is multiplicative, which explains why the signs of permutations are multiplicative.

0.1.6 Fast Fourier Transform

Let $A(x) = a_0 + a_1 x + \dots a_{n-1} x^{n-1}$ be a polinomial and let $\omega^n = 1$ be a nth root of unity. We use **FFT** to multiply two polynomials in $n \log n$ time.

Definition (DFT Matrix)— Let ω be a nth root of unity. The **DFT Matrix** is a $n \times n$ matrix given by:

$$W = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{pmatrix}$$

And its inverse is given by:

$$W^{-1} = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \dots & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \dots & \omega^{-(n-1)^2} \end{pmatrix}$$

Definition (Discrete Fourier Transform)— The **Discrete Fourier transform** (DFT) of the polynomial $A(x)$ (or equivalently the vector of coefficients $(a_0, a_1, \dots, a_{n-1})$ is defined as the values of the polynomial at the points $x = \omega_n^k$, i.e. it is the vector:

$$\text{DFT}(a_0, a_1, \dots, a_{n-1}) = (y_0, y_1, \dots, y_{n-1}) = \left(A\left(\omega_n^0\right), A\left(\omega_n^1\right), \dots, A\left(\omega_n^{n-1}\right)\right)$$

In other words, we can write $\text{DFT}(A)$ as:

$$\text{DFT}(A) = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

Definition (Inverse Discrete Fourier Transform)— The **Inverse Discrete Fourier Transform** of values of the polynomial $(y_0, y_1, \dots, y_{n-1})$ are the coefficients of the polynomial $(a_0, a_1, \dots, a_{n-1})$

$$\text{InverseDFT } (y_0, y_1, \dots, y_{n-1}) = (a_0, a_1, \dots, a_{n-1})$$

Thus, if a direct DFT computes the values of the polynomial at the points at the n -th roots, the inverse DFT can restore the coefficients of the polynomial using those values.

Algorithm (Multiplication of two polynomials)— Say we have A, B two polynomials, and we want to compute $(A \cdot B)(x)$. Then we first component-wise multiply $\mathrm{DFT}(A)$ and $\mathrm{DFT}(B)$, and then retrieve the coefficients of $A \cdot B$ by applying the Inverse DFT.

Algorithm (Fast Fourier Transform)— We want to compute the $\mathrm{DFT}(A)$ in $n \log n$ time. Suppose n is a power of 2, $\omega^n = 1$, and $A(x) = a_0 x^0 + a_1 x^1 + \cdots + a_{n-1} x^{n-1}$. We use divide and conquer by considering:

$$A_0(x) = a_0 x^0 + a_2 x^1 + \cdots + a_{n-2} x^{\frac{n}{2}-1}$$
$$A_1(x) = a_1 x^0 + a_3 x^1 + \cdots + a_{n-1} x^{\frac{n}{2}-1}$$
$$\text{and } A(x) = A_0(x^2) + x A_1(x^2)$$

After we have computed $\mathrm{DFT}(A_0) = (y_i^0)_{i=0}^{\frac{n}{2}-1}$ and $\mathrm{DFT}(A_0) = (y_i^1)_{i=0}^{\frac{n}{2}-1}$, we can compute $(y_i)_{i=0}^{n-1}$ by:

$$y_k = \begin{cases} y_k^0 + \omega^k y_k^1, & \text{for } k < \dfrac{n}{2} \\ y_k^0 - \omega^k y_k^1, & \text{for } k \geq \dfrac{n}{2} \end{cases}$$

Algorithm (Inverse FFT)— Since by definition we know $\mathrm{DFT}(A) = W \times A$, we have:

0.1.7 Problems

Problem 0.1.1 (Iran TST 2018 P1). Let $A_1, A_2, ..., A_k$ be the subsets of $\{1, 2, 3, ..., n\}$ such that for all

$$1 \le i, j \le k, \quad A_i \cap A_j \neq \varnothing$$

Prove that there are n distinct positive integers $x_1, x_2, ..., x_n$ such that for each $1 \le j \le k$:

$$\operatorname{lcm}_{i \in A_j} \{x_i\} > \operatorname{lcm}_{i \notin A_j} \{x_i\}$$

> **Solution** [elegance, a1267ab]. Pick k distinct primes $p_1, \dots, p_k$. Let x_i be the product of all p_j where $i \in A_j$. By assumption, for any j, m, some element of A_j is also in A_m, so
>
> $$\operatorname{lcm}_{i \in A_j} \{x_i\} = p_1 \cdots p_k$$
>
> because every prime is represented at least once. On the other hand, A_j^c has no element in common with A_j, so
>
> $$\operatorname{lcm}_{i \notin A_j} \{x_i\} \le \frac{p_1 \cdots p_k}{p_j}.$$
>
> Hence this works.

Remark. This solution probably came from the wish to keep all the lcm's of the left side equal, and making the right side, in some way, obviously smaller. Also we know that when dealing with lcm or gcd, it is wise to think about primes.

> **Solution** [induction]. Apply induction on either k or n. We can either build another subset of $[n]$ with the help of some primes, or we can extend one more element, and build a new x_n, which actually gives similar results to the previous solution.

Problem 0.1.2 (ISL 2016 C1). The leader of an IMO team chooses positive integers n and k with $n > k$, and announces them to the deputy leader and a contestant. The leader then secretly tells the deputy leader an n-digit binary string, and the deputy leader writes down all n-digit binary strings which differ from the leader's in exactly k positions.

For example, if $n = 3$ and $k = 1$, and if the leader chooses 101, the deputy leader would write down $001, 111$ and 100.

The contestant is allowed to look at the strings written by the deputy leader and guess the leader's string. What is the minimum number of guesses (in terms of n and k) needed to guarantee the correct answer?

> **Solution** [dgrozev, blogpost]. Think about the problems in terms of "Hamming" distance. Then we need to find the center of a sphere of radius k. And what's the

nicest thing with center's and points on the sphere? "Diameters". So we try to find the points on this sphere with the greatest distance, and we hope to find something about the center from the information.

Problem 0.1.3 (ISL 2005 C2). Let $a_1, a_2, \ldots$ be a sequence of integers with infinitely many positive and negative terms. Suppose that for every positive integer n the numbers $a_1, a_2, \ldots, a_n$ leave n different remainders upon division by n. Prove that every integer occurs exactly once in the sequence $a_1, a_2, \ldots$.

Solution. Constructing the initials.

Problem 0.1.4 (IOI Practice 2017). C plays a game with A and B. There's a room with a table. First C goes in the room and puts 64 coins on the table in a row. Each coin is facing either heads or tails. Coins are identical to one another, but one of them is cursed. C decides to put that coin in position c. Then he calls in A and shows him the position of the cursed coin. Now he allows A to flip some coins if he wants (he can't move any coin to other positions). After that A and C leave the room and sends in B. If B can identify the cursed coin then C loses, otherwise C wins.

The rules of the game are explained to A and B beforehand, so they can discuss their strategy before entering the room. Find the minimum number k of coin flips required by A so that no matter what configuration of 64 coins C gives them and where he puts the cursed coin, A and B can win with A flipping at most k coins.

Find constructions for $k = 32, 8, 6, 3, 2, 1$

Solution. XOR XOR XOR binary representation

Problem 0.1.5 (Codeforces 987E). Petr likes to come up with problems about randomly generated data. This time problem is about random permutation. He decided to generate a random permutation this way: he takes identity permutation of numbers from 1 to n and then $3n$ times takes a random pair of different elements and swaps them. Alex envies Petr and tries to imitate him in all kind of things. Alex has also come up with a problem about random permutation. He generates a random permutation just like Petr but swaps elements $7n+1$ times instead of $3n$ times. Because it is more random, OK?!

You somehow get a test from one of these problems and now you want to know from which one.

Solution. This theorem kills this problem instantly.

Problem 0.1.6 (USAMO 2013 P6). At the vertices's of a regular hexagon are written six nonnegative integers whose sum is 2003^{2003}. Bert is allowed to make moves of the following form: he may pick a vertex and replace the number written there by the absolute value of the difference between the numbers written at the two neighboring vertices. Prove that Bert can make a sequence of moves, after which the number 0 appears at all six vertices.

> **Solution.** Firstly what comes into mind is to decrease the maximum value, but since this is a P6, there must be some mistakes. Surely, we can't follow this algo in the case $(k,k,0,k,k,0)$. But this time, the sum becomes even. So we have to slowly minimize the maximum, keeping the sum odd. And since only odd number on the board is the easiest to handle, we solve that case first, and the other cases can be easly handled with an additional algo.

Problem 0.1.7 (ISL 2012 C1). Several positive integers are written in a row. Iteratively, Alice chooses two adjacent numbers x and y such that $x > y$ and x is to the left of y , and replaces the pair (x,y) by either $(y+1,x)$ or $(x-1,x)$. Prove that she can perform only finitely many such iterations.

> **Solution.** Easy invariant.

Problem 0.1.8 (AoPS). There is a number from the set $\{1,-1\}$ written in each of the vertices's of a regular do-decagon (12 -gon). In a single turn we select 3 numbers going in the row and change their signs. In the beginning all numbers, except one are equal to 1. Can we transfer the only -1 into adjacent vertex after a finite number of turns?

> **Solution.** Algo+Proof $\Longrightarrow$ Invariant.

Problem 0.1.9 (ISL 1994 C3). Peter has three accounts in a bank, each with an integral number of dollars. He is only allowed to transfer money from one account to another so that the amount of money in the latter is doubled. Prove that Peter can always transfer all his money into two accounts. Can Peter always transfer all his money into one account?

> **Solution.** Since we want to decrease the minimum, and one of the most simple way is to consider Euclidean algorithm. So we sort the accounts, $A < B < C$, and write $B = qA + r$, and do some experiment to turn B into r.

Problem 0.1.10 (MEMO 2008, Team, P6). On a blackboard there are $n \geq 2, n \in \mathbb{Z}^+$ numbers. In each step we select two numbers from the blackboard and replace both of them by their sum. Determine all numbers n for which it is possible to yield n identical number after a finite number of steps.

Solution. The pair thing rules out the case of odds. For evens, we make two identical sets, and focus on only one of the sets, with an additional move $x \to 2x$ available to use. Since we can now change the powers of 2 at out will at any time, we only focus on the greatest odd divisors. Our aim is to slowly decrease the largest odd divisor.

Problem 0.1.11 (USA Dec TST 2016, P1). Let $S = \{1, \ldots, n\}$. Given a bijection $f : S \to S$ an orbit of f is a set of the form $\{x, f(x), f(f(x)), \ldots\}$ for some $x \in S$. We denote by $c(f)$ the number of distinct orbits of f. For example, if $n = 3$ and $f(1) = 2$, $f(2) = 1$, $f(3) = 3$, the two orbits are $\{1, 2\}$ and $\{3\}$, hence $c(f) = 2$.

Given k bijections f_1, ..., f_k from S to itself, prove that

$$c(f_1) + \cdots + c(f_k) \leq n(k-1) + c(f)$$

where $f : S \to S$ is the composed function $f_1 \circ \cdots \circ f_k$.

Solution. Induction reduces the problem to the case of $k = 2$. Then another inuction on $c(f_1)$ solves the problem. The later induction works on the basis of the fact that a "swap" in the bijection changes the number of cycles by 1 (either adds $+1$ or -1).

Problem 0.1.12 (Cody Johnson). Consider a set of 6 integers $S = \{a_1 \ldots a_6\}$. At on step, you can add $+1$ or -1 to all of the 6 integers. Prove that you can make a finite number of moves so that after the moves, you have $a_1 a_5 a_6 = a_2 a_4 a_6 = a_3 a_4 a_5$

Problem 0.1.13 (ISL 2014 A1). Let $a_0 < a_1 < a_2 \ldots$ be an infinite sequence of positive integers. Prove that there exists a unique integer $n \geq 1$ such that

$$a_n < \frac{a_0 + a_1 + a_2 + \cdots + a_n}{n} \leq a_{n+1}.$$

Solution. My idea was to construct the sequence with the assumtion that the condition is false. It leads to either all of the right ineq false or the condition being true.

Solution. The magical solution: defining $b_n = (a_n - a_{n-1}) + \cdots + (a_n - a_1)$ which eases the inequlity.

Solution. The beautiful solution: defining δ_i as $a_n = a_0 + \Delta_1 + \Delta_2 + \ldots + \Delta_n$ for all n, i.

> **Solution.** Another idea is to first prove the existence and then to prove the uniqueness.

Problem 0.1.14 (ISL 2014 N3). For each positive integer n, the Bank of Cape Town issues coins of denomination $\frac{1}{n}$. Given a finite collection of such coins (of not necessarily different denominations) with total value at most most $99+\frac{1}{2}$, prove that it is possible to split this collection into 100 or fewer groups, such that each group has total value at most 1.

> **Solution.** Notice that the sum of the geometric series $S=\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}\dots$ is 1. And in another problem we partitioned the set of integers into subsets with each subset starting with an odd number k and every other elements of the subset being 2^i*k. We do similarly in this problem, and partition the set of the coins in a similar way. Then we take the first 100 sets whose sum is less than 1 and insert the other left coins in these sets, with the condition that the sum of all of the coins is $99+\frac{1}{2}$. Solu

> **Solution.** Replacing 100 by n, we show that for all n the codition is valid. Assume otherwise. Take the minimal n for which the condition does not work. Ta-Da! We can show that if n does not work, so doesn't $n-1$. Solu

> **Solution.** Or just be an EChen and prove the result for at most $k-\frac{k}{2k+1}$ with k groups.

> **Solution.** Very similar to this problem

Problem 0.1.15 (China TST 2006). Given positive integer n, find the biggest real number C which satisfy the condition that if the sum of the reciprocals ($\frac{1}{n}$ is the reciprocal of n) of a set of integers (They can be the same.) that are greater than 1 is less than C, then we can divide the set of numbers into no more than n groups so that the sum of reciprocals of every group is less than 1.

Problem 0.1.16. In a $n*n$ grid, every cell is either black or white. A 'command' is a pair of integers, $i,j\leq n$, after which all of the cells in the i^{th} row and the j^{th} column (meaning a total of $2n-1$ cells) will switch the state. Our goal is to make every cell of the same state.

1. Prove that if it can be done, it can be done in less than $\frac{n^2}{2}$ commands.

2. Prove that it can always be done if n is even.
3. Prove or disprove for odd n.

> **Solution.** (a) is really easy, just take into account that flipping all cells result in the switch of all of the cells. And the question did not ask for an algorithm.

> **Solution.** (b) is also easy, notice that we can pair the columns and then make them look like the same, with a compound command. A better algo is to take the original algo and to modify it like, take one cell, then do the original move on all cells in the row and column of this cell.

> **Solution.** (c) uses Linear Algebra, which I dont know yet, or... use double counting to build the criteria of the fucntion f : states $\to$ subset of moves being bijective.

Problem 0.1.17 (OIM 1994). In every square of an $n \times n$ board there is a lamp. Initially all the lamps are turned off. Touching a lamp changes the state of all the lamps in its row and its column (including the lamp that was touched). Prove that we can always turn on all the lamps and find the minimum number of lamps we have to touch to do this.

Problem 0.1.18 (AtCoder GC043 B). Given is a sequence of N digits $a_1, a_2 \ldots a_N$, where each element is $1, 2$, or 3. Let $x_{i,j}$ defined as follows:

- $x_{1,j} := a_j \quad (1 \leq j \leq N)$
- $x_{i,j} := |x_{i-1,j} - x_{i-1,j+1}| \quad (2 \leq i \leq N \text{ and } 1 \leq j \leq N+1-i)$

Find $x_{N,1}$.

> **Solution.** Since $|x-y| \equiv x+y \pmod 2$, we can determine the parity of $X_{N,1}$ using binomial coefficient. Which in turn we can get in $O(n)$ with bitwise operator. Now we have to distinguish between $0, 2$. For 2, all of the rows starting with the second one should have only 2 and 0. Where we can apply the same algorithm as before and find whether the final digit is 2 or 0.

Problem 0.1.19 (Timus 1578). The very last mammoth runs away from a group of primeval hunters. The hunters are fierce, hungry and are armed with bludgeons and stone axes. In order to escape from his pursuers, the mammoth tries to foul the trail. Its path is a polyline (not necessarily simple). Besides, all the pairs of adjacent segments of the polyline form acute angles (an angle of 0 degrees is also considered acute).

After the mammoth vanished, it turned out that it had made exactly N turns while running away. The points where the mammoth turned, as well as the points where the pursuit started and where the pursuit ended, are known. You are to determine one if the possible paths of the mammoth.

Problem 0.1.20 (Codeforces 744B). Given a hidden matrix of $n \times n$, $n \leq 1000$ where for every i , $M_{(i,i)=0}$, Luffy's task is to find the minimum value in the n rows, formarly spoken, he has to find values $\min_{j=1\ldots n,\ j\neq i} M_{(i,j)}$. To do this he can ask the computer questions of following types: In one question, Luffy picks up a set, $a_1, a_2 \ldots a_k$ with $a_i, k \leq n$. And gives the computer this set. The computer will respond with n integers. The i -th integer will contain the minimum value of $\min_{j=1\ldots k} M_{(i,a_j)}$. And on top of this, he can only ask 20 questions. Luffy being the stupid he is, doesn't even have any clue how to do this, you have to help him solving this problem.

Solution. If we draw the diagonal in the matrix, we see that we can fit boxes of $2^i \times 2^i$ in there depending on the i 's value. Now after we have decomposed the matrix into such boxes, we can choose several from them to ask a question. The trick is that for every row, there must be questions asked from each of the boxes this row covers and no question from here must contain the (i, i) cell.

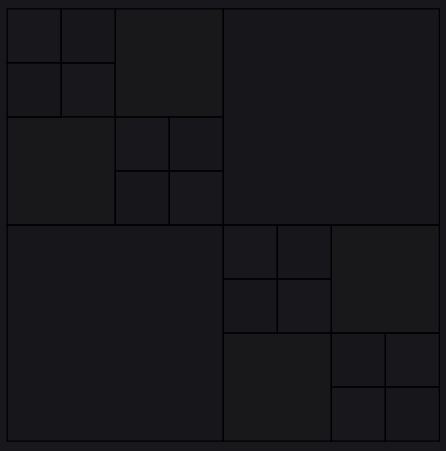

Figure 0.3: haha

Solution [magical solution]**.** For $i \leq 10$, for every $k = 1 \ldots n$, include k in the question if the ith bit of k's binary form is 0. And then for the second round include k in the question if the i th bit of k's binary form is 1.

Problem 0.1.21. Alice wants to add an edge (u, v) in a graph. You want to know what this edge is. So, you can ask some questions to Alice. For each question, you will give Alice 2 non-empty disjoint sets S and T to Alice, and Alice will answer "true" iff u and v belongs to different sets. You can ask atmost $3 * \lceil log_2 |V| \rceil$ questions to Alice. Describe a strategy to find the edge (u, v).

Solution. First find one true answer in $\lceil log_2 |V| \rceil$ questions, and then get the result out of these two sets in $2 * \lceil log_2 |V| \rceil$ questions.

> **Solution.** The magical solution goes as following: In the i^{th} question, $S = x : i^{th}$ bit of x is 0, $T = x : i^{th}$ bit of x is 1

Problem 0.1.22 (USAMO 2015 P4). Steve is piling $m \geq 1$ indistinguishable stones on the squares of an $n \times n$ grid. Each square can have an arbitrarily high pile of stones. After he finished piling his stones in some manner, he can then perform stone moves, defined as follows. Consider any four grid squares, which are corners of a rectangle, i.e. in positions $(i,k),(i,l),(j,k),(j,l)$ for some $1 \leq i,j,k,l \leq n$, such that $i < j$ and $k < l$. A stone move consists of either removing one stone from each of (i,k) and (j,l) and moving them to (i,l) and (j,k) respectively,j or removing one stone from each of (i,l) and (j,k) and moving them to (i,k) and (j,l) respectively.

Two ways of piling the stones are equivalent if they can be obtained from one another by a sequence of stone moves.

How many different non-equivalent ways can Steve pile the stones on the grid?

> **Solution.** Building an invariant, we see that only the sum of the columns is not sufficient. So to get more control, we take the row sums into account as well.

Problem 0.1.23 (ISL 2003 C4). Let $x_1,\ldots,x_n$ and $y_1,\ldots,y_n$ be real numbers. Let $A = (a_{ij})_{1\leq i,j\leq n}$ be the matrix with entries

$$a_{ij} = \begin{cases} 1, & \text{if } x_i + y_j \geq 0; \\ 0, & \text{if } x_i + y_j < 0. \end{cases}$$

Suppose that B is an $n \times n$ matrix with entries 0, 1 such that the sum of the elements in each row and each column of B is equal to the corresponding sum for the matrix A. Prove that $A = B$.

> **Solution.** If done after this problem, this problem seems straigthforward.

Problem 0.1.24 (India TST 2017 D1 P3). Let $n \geq 1$ be a positive integer. An $n \times n$ matrix is called *good* if each entry is a non-negative integer, the sum of entries in each row and each column is equal. A *permutation* matrix is an $n \times n$ matrix consisting of n ones and $n(n-1)$ zeroes such that each row and each column has exactly one non-zero entry.

Prove that any *good* matrix is a sum of finitely many *permutation* matrices.

> **Solution.** Same algo as above. Either distributing uniformly or gathering all in a diagonal

Problem 0.1.25 (Tournament of Towns 2015F S7). N children no two of the same height stand in a line. The following two-step procedure is applied: first, the line is split into the least possible number of groups so that in each group all children are arranged from the left to the right in ascending order of their heights (a group may consist of a single child). Second, the order of children in each group is reversed, so now in each group the children stand in descending order of their heights. Prove that in result of applying this procedure $N-1$ times the children in the line would stand from the left to the right in descending order of their heights.

> **Solution.** It's obvious that we need to find some invariant or mono-variant. Now, an idea, we need to show that for any i, for it to be on it's rightful place, it doesn't need more than $N-1$ moves. How do we show that? Another idea, think about the bad bois on either of its sides. Now, observation, 'junctions' decrease with each move. Find the 'junctions'.

Problem 0.1.26 (Polish OI). Given n jobs, indexed from $1,2\ldots n$. Given two sequences of reals, $\{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n$ where, $0 \leq a_i, b_i \leq 1$. If job i starts at time t , then the job takes $h_i(t) = a_i t + b_i$ time to finish. Order the jobs in a way such that the total time taken by all of the jobs is the minimum.

> **Solution.** Example of a problem which is solved by investigating two adjacent objects in the optimal arrangement.

Problem 0.1.27 (Codeforces 960C). Pikachu had an array with him. He wrote down all the non-empty subsequences of the array on paper. Note that an array of size n has $2^n - 1$ non-empty subsequences in it.

Pikachu being mischievous as he always is, removed all the subsequences in which

$$\text{Maximum element of the subsequence} - \text{Minimum element of subsequence} \geq d$$

Pikachu was finally left with X subsequences.

However, he lost the initial array he had, and now is in serious trouble. He still remembers the numbers X and d. He now wants you to construct any such array which will satisfy the above conditions. All the numbers in the final array should be positive integers less than 10^{18}.

Note the number of elements in the output array should not be more than 10^4. If no answer is possible, print -1.

Problem 0.1.28 (ARO 2005 P10.3, P11.2). Given 2005 distinct numbers $a_1, a_2, \ldots, a_{2005}$. By one question, we may take three different indices $1 \leq i < j < k \leq 2005$ and find out the set of numbers $\{a_i, a_j, a_k\}$ (unordered, of course). Find the minimal number of questions, which are necessary to find out all numbers a_i.

> **Solution.** We want to maximize the amount of information each query gives us. So we need to tie our questions with our previous questions to find as many integers as possible. This leads us to the greedy approach where we circle around the positions.

Problem 0.1.29 (IOI 2007 P3). You are given two sets of integers $A = \{a_1, a_2 \ldots a_n\}$ and $B = \{b_1, b_2 \ldots b_n\}$ such that $a_i \geq b_i$. At move i you have to pick b_i distinct integers from the set $A_i = \{1, 2, \ldots a_i\}$. In total, $(b_1 + b_2 + \cdots + b_n)$ integers are selected, but not all of these are distinct.

Suppose k distinct integers have been selected, with multiplicities $c_1, c_2, c_3 \ldots c_k$. Your score is defined as

$$\sum_{i=1}^{k} c_i(c_i - 1)$$

Give an efficient algorithm to select numbers in order to "minimize" your score.

> **Solution.** Some investigation shows that if $c_i > c_j + 1$ and $i > j$, then we can always minimize the score. and if $i < j$, then we can minimize the score only when $i, j \in A_k$ but i has been taken at move k , but j hasn't.
>
> So in the minimal state, either both i, j has been taken at move k , or $a_k < j$. So the idea is to take elements from A_i as large as possible, and then taking smaller values after wards if the c_i value of a big element gets more than that of a small element. In this algorithm, we see that we greedily manipulate c_i. So it is a good idea to greedily choose c_i 's from the very beginning.
>
> At step i, take the set $\{c_1, c_2 \ldots c_{a_i}\}$ and take the smallest b_i from this set, and add 1 to each of them (in other words, take their index numbers as the numbers to take).

Problem 0.1.30. Given n numbers $\{a_1, a_2, ..., a_n\}$ in arbitrary order, you have to select k of them such that no two consecutive numbers are selected and their sum is maximized.

> **Solution.** Notice that if a_i is the maximum value, and if a_i is not counted in the optimal solution, then both of a_{i-1}, a_{i+1} must be in the optimal solution, and $a_{i-1} + a_{i+1} > a_i$. And if a_i is counted in the optimal solution, then none of a_{i-1}, a_{i+1} can be counted in the optimal solution. So either way, we can remove these three and replace them by a single element to use induction. So remove a_{i-1}, a_{i+1} and replace a_i by $a_{i-1} + a_{i+1} - a_i$.

Problem 0.1.31 (USAMO 2010 P2). There are n students standing in a circle, one behind the other. The students have heights $h_1 < h_2 < \cdots < h_n$. If a student with height h_k is standing directly behind a student with height h_{k-2} or less, the two students are permitted to switch places. Prove that it is not possible to make more than $\binom{n}{3}$ such switches before reaching a position in which no further switches are possible.

> **Solution.** Instead of just thinking about the switches randomly, we try to count the number a_k of times when the student k switched his position. Trying out some smaller cases with this, we discover the upper bound of each a_k.

Problem 0.1.32 (Serbia TST 2015 P3). We have 2015 prisinoers. The king gives everyone a hat colored in one of 5 colors. Everyone sees others hats expect his own.

Now, the King orders them in a line (a prisioner can see all guys behind and in front of him). The King asks the prisinoers one by one if he knows the color of his hat.

If he answers NO, then he is killed. If he answers YES, then he is asked the color of his hat. If his answer is true, he is freed, if not, he is killed. All the prisinors can hear his YES-NO answer, but if he answered YES, they don't know what he answered (he is killed in public).

They can think of a strategy before the King comes, but after that they can't communicate. What is the largest number of prisoners they can guarentee to survive?

> **Solution** [MellowMelon]**.** Since everyone can see other's caps and hear other's answers, so we can try global approach. Usually in these "encryption" problems, we want to use some bits to express an integer that has all the information needed.
>
> In this case, if everyone knows the sum of others cap numbers, then they can easily find their's. They don't even need to know the sum, they can do it with the sum modulo 5. This idea gives us the answer 2012, because we need at least 3 bits to express all 5 cases.
>
> But wait, what about the YES answer and then getting killed or being freed? We didn't use that at all! It turns out we can use those extra informations to save one more person!
>
> Let A be #2's hat mod 5 and let B be the sum of #3 through #2015 mod 5. The goal is for everyone to know B after #1 and #2 take their turns.
>
> If $A + B \equiv 0, 1 \pmod 4$ or $(A, B) = (2, 2)$ or $(A, B) = (0, 4)$, then #1 says no. Otherwise, #1 says yes and guesses whatever. Let $X = 0$ if #1 said no and $X = 1$ if #1 said yes.
>
> If #2 sees $B = 0$, (s)he says no.

If #2 sees $B = 1$, (s)he says yes and guesses hat color $3X$ mod 5.
If #2 sees $B = 2$, (s)he says no.
If #2 sees $B = 3$, (s)he says yes and guesses hat color $3 + 3X$ mod 5.
If #2 sees $B = 4$, (s)he says yes and guesses hat color $2 + 2X$ mod 5.

If you draw out the five by five grid with all values of A, B and all results of the above, you'll see that the results of #1's and #2's guesses uniquely identify B in all cases.

Problem 0.1.33 (ISL 2014 N1). Let $n \geq 2$ be an integer, and let A_n be the set

$$A_n = \{2^n - 2^k \mid k \in \mathbb{Z}, 0 \leq k < n\}.$$

Determine the largest positive integer that cannot be written as the sum of one or more (not necessarily distinct) elements of A_n .

Solution. We know similar problem from chicken mcnugget theorem, which motivates us to try out some smaller cases. And those cases further motivates us to formulate a conjecture. After the answer has been found, it is not hard to prove that, again by recursively trying to build the sum.

Solution [getrektm9, motivation for the answer]**.** My idea for guessing the answer was observing modulo 2^{n-1}. I wasn't trying to find the smallest numeber that we can represent as sum of elements from the set with the condition that it is congruent to modulo 1 while dividing with 2^{n-1}. It turns out that the smallest number that we can't represent is exatcly $(n-2) \cdot 2^n + 1$.

The reason for observing modulo 2^{n-1} is that the smallest number in the set is 2^{n-1} and later showed that every other residue bigger than 1 can be obtained with the condition that its smallest sum is less than when residue is 1.

Problem 0.1.34 (Taiwan TST 2015 R3D1P1). A plane has several seats on it, each with its own price, as shown below. $2n-2$ passengers wish to take this plane, but none of them wants to sit with any other passenger in the same column or row. The captain realize that, no matter how he arranges the passengers, the total money he can collect is the same. Proof this fact, and compute how much money the captain can collect.

					n	n					
				n	n-1	n-1	n				
			n	n-1	n-2	n-2	n-1	n			
		⋰	⋰	⋮	⋮	⋮	⋮	⋱	⋱		
	n	…		…	3	3	…		…	n	
n	n-1	…		3	2	2	3		…	n-1	n
n-1	…	…		2	1	1	2		…	…	n-1
…	…		2	1			1	2		…	…
⋮	⋮									⋮	⋮
3	2	1							1	2	3
2	1									1	2
1											1

Table 0.1: Problem 0.1.34, there are $2n-2$ columns and $2n-2$ rows.

Solution [induction]**.** This table looks really appropriate for a nice induction approach. We want to prove that the total is n^2-1, which tells us how much change we will get by going from n to $n-1$. And the seats that will guarantee that this change will occure are the one in the first row, and the two seats in the last and first columns. But we need to do some work to get them nicely in position.

We use the usual grid movement trick: $(a,b),(c,d) \leftrightarrow (a,d),(c,b)$.

Solution [MellowMelon]**.** The price of the seats in this arrangement are almost like the pricing of the lattice points by $x+y$. And if we give the rows and columns their own prices and change the pricing system like so, we get an easy invariant.

Problem 0.1.35 (`Codeforces 330D`)**.** Biridian Forest

Solution. Generalize the condition for a meet-up.

Problem 0.1.36 (`Codeforces 1270F`)**.** Let's call a binary string s awesome, if it has at least 1 symbol 1 and length of the string is divisible by the number of 1 in it. In particular, `1`, `1010`, `111` are awesome, but `0`, `110`, `01010` aren't.

You are given a binary string s of size $\leq 2\times 10^5$. Count the number of its awesome substrings.

Solution. The constraint tells us the algorithm should be of complexity $O(n\sqrt{n})$. Playing with the funtion $f(i)$ and the divisibility condition gives us some ground to work with.

Since we need $\leq c \times \sqrt{n}$ queries around the full string, we know we need to use something similar to the prime sieve trick.

Problem 0.1.37 (IMO 1986 P3). To each vertex of a regular pentagon an integer is assigned, so that the sum of all five numbers is positive. If three consecutive vertices are assigned the numbers x, y, z respectively, and $y < 0$, then the following operation is allowed: x, y, z are replaced by $x+y, -y, z+y$ respectively.

Such an operation is performed repeatedly as long as at least one of the five numbers is negative. Determine whether this procedure necessarily comes to an end after a finite number of steps.

Solution [invariant]**.** The move changes a lot except the sum. And if we wanted it to terminate at some point, we know there might be an invariant that decreases monotonically. What is the closest form of invariance to sum? Sum of squares. But sum of squares doesn't tell us much either. But we know sometimes sum of squares of differences bear good results. So we try some of the different functions we get from there.

And voila, if we let

$$f(a_1, a_2, a_3, a_4, a_5) = \sum_{i=1}^{5} (a_i - a_{i+2})^2$$

we get a value that decreases!

Solution. Notice how starting from one negative number and applying the move to the negative number created to its one side and keeping on doing this, we can see the numbers move to the other direction. We use this idea to come up with a compound move to increase the least number.

Problem 0.1.38 (Codeforces 1379E). Let G be a tree with n vertices that has a root, and every vertex has either 2 children or no child. A vertex is said to be "good" if the two subtrees of it, say X, Y, satisfy either $V(X) \geq 2V(Y)$ or $V(Y) \geq 2V(X)$ (here $V(X)$ is the number of vertices of tree X). Find all k such that there is a tree with n vertices with k good vertices, and give a construction to build the tree.

Solution. The first thing we notice is the for even n, no such tree exists. Now, binary trees are really nice, because we can build them recursively: Delete the root, and build the two subtrees by recursion. Trying out for some initial values, we get the solution

pattern that if $S(n)$ is the set of possible values for k, then

$$S(n) = \begin{cases} \left\{1, 2, \dots \frac{n-3}{2}\right\} & \text{if } n \neq 2^t - 1 \\ \left\{0, 2, \dots \frac{n-3}{2}\right\} & \text{if } n = 2^t - 1 \end{cases}$$

Which isn't hard to proof by induction.

Problem 0.1.39 (ISL 2010 C4). Each of the six boxes B_1, B_2, B_3, B_4, B_5, B_6 initially contains one coin. The following operations are allowed

1. Choose a non-empty box B_j, $1 \leq j \leq 5$, remove one coin from B_j and add two coins to B_{j+1};
2. Choose a non-empty box B_k, $1 \leq k \leq 4$, remove one coin from B_k and swap the contents (maybe empty) of the boxes B_{k+1} and B_{k+2}.

Determine if there exists a finite sequence of operations of the allowed types, such that the five boxes B_1, B_2, B_3, B_4, B_5 become empty, while box B_6 contains exactly $2010^{2010^{2010}}$ coins.

Solution. The given number is absurdly big, with no apparent reason right now, so we decide to just look for a way to maximize a box as much as we can.

After playing around with this idea, we soon notice that we can make powers of 2, with some compound moves. Which in turn provides the construction for nested powers of twos. So it might not be impossible to get a large enough nested power of two that is greater than our large number.

But what about "exactly" condition? The only move that allows us to reduce is the second move, which requires two boxes at the end, So we need to make the big number appear somewhere in B_1, B_2, B_3 or B_4.

0.1.8 Algorithm Analysis

Problem 0.1.40 (GQMO 2020 P4). Prove that, for all sufficiently large integers n, there exists n numbers $a_1, a_2, \ldots, a_n$ satisfying the following three conditions:

Each number a_i is equal to either $-1, 0$ or 1. At least $\frac{2n}{5}$ of the numbers $a_1, a_2, \ldots, a_n$ are non-zero. The sum $\frac{a_1}{1} + \frac{a_2}{2} + \cdots + \frac{a_n}{n}$ is 0.

Note: Results with 2/5 replaced by a constant c *will be awarded points depending on the value of* c

0.1.9 Covering Area with Squares

- A nice blog post by ankogonit

Problem 0.1.41 (Brazilian MO 2002, ARO 1979). Given a finite collection of squares with total area at least 4, prove that you can cover a unit square completely with these squares (with overlapping allowed, of course).

> **Solution.** Maybe motivated by the number 4 and how nice it would be if all the squares had 2's power side lengths, the idea is to shrink every square to a side with side of 2's power.

Problem 0.1.42 (ARO 1979's Sharper Version). Given a finite collection of squares with total area at least 3, prove that you can cover a unit square completely with these squares (with overlapping allowed).

> **Solution.** The idea is to greedily cover the unit square by covering the lowest row uncovered. And then using boundings to prove that it is possible.

Problem 0.1.43. Prove that a finite collection of squares of total area $\frac{1}{2}$ can be placed inside a unit square without overlap.

> **Solution.** The same idea as before.

Problem 0.1.44 (Tournament of Towns Spring 2012 S7). We attempt to cover the plane with an infinite sequence of rectangles, overlapping allowed.

1. Is the task always possible if the area of the $n-$th rectangle is n^2 for each n?
2. Is the task always possible if each rectangle is a square, and for any number N, there exist squares with total area greater than N?

> **Solution.** Identical algo and proving technique as above.

> **Solution.** Using the first problem in this subsection to find a better algo.

Problem 0.1.45 (ISL 2006 C6). A holey triangle is an upward equilateral triangle of side length n with n upward unit triangular holes cut out. A diamond is a $60^\circ - 120^\circ$ unit

rhombus.

Prove that a holey triangle T can be tiled with diamonds if and only if the following condition holds: Every upward equilateral triangle of side length k in T contains at most k holes, for $1 \leq k \leq n$.

> **Solution.** Think of induction and how you can deal with that.

Problem 0.1.46 (Putnam 2002 A3). Let N be an integer greater than 1 and let T_n be the number of non empty subsets S of $\{1,2,.....,n\}$ with the property that the average of the elements of S is an integer. Prove that $T_n - n$ is always even.

> **Solution.** Try to show an bijection between the sets with average n which has k elements (Here k is an even integer) and the sets with average n but with number of elements $k+1$. This implies that the number of such sets is even.

Problem 0.1.47 (USAMO 1998). Prove that for each $n \geq 2$, there is a set S of n integders such that $(a-b)^2$ divides ab for every distinct $a, b \in S$.

> **Solution.** Induction comes to the rescue. Trying to find a way to get from n to $n+1$, we see that we can *shift* the integers by any integer k. So after shifting, what stays the same, and what changes?

Problems

Theorems

Definitions

Strategies